Twins Vertices in Hypergraphs

Raoul Medina[1] and Caroline Noyer and
Olivier Raynaud[2]

Research Report LIMOS/RR-06-07

24 juillet 2006

[1]medina@isima.fr
[2]raynaud@isima.fr

Abstract

Twin vertices in graphs correspond to vertices sharing the same neighborhood. We propose an extension to hypergraphs of the concept of twin vertices. For this we give two characterizations of twin vertices in hypergraphs, a first one in term of clone vertices (the concept of clone has been introduced in [16]) and a second one in term of committees (introduced in [6]). Finally we give an algorithm to aknowledge a set as committee and two algorithms to compute clone-twin vertices classes ans committee-twin vertices classes.

Keywords: Hypergraphs, problem of generation, symmetries, twin vertices

1 Introduction

An important research area in computer science focus on the generation of combinatorics objects from a discrete structure. As example the extraction of cliques or stables of a graph, maximal antichains or ideals from an order ([15, 18]) and minimal implication base from a closure system ([10]). One major difficulty of this kind of problems lies in the fact that the size of the result is often potentially exponential. A known approach is to check for symmetries in the instances before any computation. These symmetries allow to represent the instance, as well as the result, in a more compact form. Whenever two vertices play a symmetrical role in the discrete structure as well as in the generated combinatorics objects, the generation process can be done on a reduced instance (by removing symmetrical objects in the discrete structure). Reconstructing the result of the original instance from the results obtained with the reduced instance is then easily done by computing the missing symmetrical objects.

Known forms of symmetries in graphs are, for example, twin vertices (i.e. vertices with the same neighborhood) or modules (sets of vertices having same neighborhood outside the module). One can notice that twin vertices define modules of size two in a graph. These symmetries admit immediate applications. For instance, computing cliques from a graph could be done from a graph without any twin vertices. Indeed, twin vertices belong to the same set of cliques. Definition of modules allows a tree decomposition process of a graph in different types of modules (parallel or serie nodes). This modular decomposition allows to apply algorithms adapted to the type of the module. It also allows to characterize particular classes of graphs : the prime graphs being graphs with only single vertex modules.

Some symmetries in graphs find natural generalization in hypergraphs. For instance, the concept of module of a graph has been extended to hypergraphs under the term of committee ([4, 6, 7]) or module of an hypergraph ([6, 8]). Note that in this generalization, a module is just a particular case of committees. In this paper our goal is to propose an extension of the concept of twin vertices for a couple of vertices of an hypergraph. Such extension should respect the symmetry in the hypergraph as well as to be of algorithmic interest when confronted to the problem of generating combinatorial objects from the hypergraph. We will illustrate our proposed extensions on the problem of generating *k-hypercliques* of an *k-hypergraph*. In particular, we will see that defining twin vertices in hypergraphs using an extension of the neighborhood relation in graphs does not respect the symmetries in *k-hypercliques*. Thus, we will see how to characterize twin vertices in hypergraph with the notion of

clone elements in a collection of sets. This concept of clone has been introduced in [16]. We will show that clone elements also characterize twin vertices in graphs. We will also use the committee definition of [4, 6, 7] as an alternative to define twin vertices of hypergraphs. For each proposed definition, we will give algorithms to detect the corresponding twin vertices in an hypergraph.

This paper is organized as follows. In section 2, we recall definitions of twin vertices and modules in graphs as well as a characterization of twin vertices in terms of modules. In the third section we propose two different generalizations of twin vertices in hypergraph. A first one in term of clone vertices (clone-twins) and a second in term of committees (committee-twins). Then we compare properties of each of these characterizations. In the fourth section we give an algorithm to compute classes of clone-twin vertices of an hypergraph and an algorithm to check if a given set forms a committee of an hypergraph. Reader will find some demonstrations of our results in an appendice.

In this paper we will use the following notation : cursive letters as $\mathcal{E}$ or $\mathcal{F}$ will denote families of sets, capital letters as $(V, E, F, X...)$ will denote sets and small letters as $(a, b, ...)$ will denote elements of a set.

2 Twin vertices in graphs

Definition 1 *Let $G = (V, E)$ be a graph and v a vertex of V, we call neighborhood of v and we note $\mathcal{V}(v)$ the set of vertices y such that the edge (v, y) belongs to E. The closed neighborhood, noted $\mathcal{V}[v]$, is equal to $\mathcal{V}(v) \cup \{v\}$.*

Definition 2 *Let $G = (V, E)$ be a graph, we say that two vertices a and b are twin vertices in G* ***if and only if*** *they have the same neighborhood. In addition, if the edge (a, b) belongs to E, a and b are said true twins (characterized by $\mathcal{V}[a] = \mathcal{V}[b]$), if not they are said false twins (characterized by $\mathcal{V}(a) = \mathcal{V}(b)$). Twin relations between vertices of a graph are equivalence relation.*

One application is to compute classes of twin vertices of a graph before computing some combinatorics objects on the graph. For example, consider a and b twin vertices of a graph G and C a clique of G such that a belongs to C.

- If a and b are true twins, then the set $C \cup \{b\}$ is also a clique of G ;
- If a and b are false twins, then the set $C \setminus \{a\} \cup \{b\}$ is also a clique of G.

For this reason, computing cliques of a graph is often done in four steps : first compute the class of twin vertices, then reduce the graph G by deleting from V all vertices but one of each class. In a third step compute the set of cliques of the new graph. Finally compute the missing symmetrical cliques using the information about classes of (true or false) twin vertices of G.

Another interest of the twin symmetry is to capture characteristics of some classes of graphs. Let us focus on the class of *distance-hereditary* graphs. Originally defined as graphs in which every induced path is isometric, these graphs admit many others characterizations. One of them in term of twin vertices (see [2] for a survey). Thanks to this characterization authors of [12] give a linear time and space algorithm to compute the covering by complete bipartite subgraphs for *distance-hereditary* graphs. This problem being NP-Hard in the general case (Problem GT18 in [9]). A original demonstration of this last result is given in [11].

In the following we define modules of a graph and use it to characterize twin vertices of a given graph.

Definition 3 *Let $G = (V, E)$, we say that $A \subseteq V$ is a module of G* ***if and only if*** *for all couples (x, y) of A^2, the sets $\mathcal{V}(y) \setminus A$ and $\mathcal{V}(b) \setminus A$ are equal.*

In other words, vertices of a module have same neighborhood outside the module.

One can trivially restate the twin vertices definition using modules.

Definition 4 *Let $G = (V, E)$ be a graph, then a and b in V are twin vertices of G if the set $\{a, b\}$ is a module of G.*

It is known that a partition of a graph in disjoint module sets leads to a tree representation of the graph. Indeed this tree corresponds to the recursive schema of decomposition of the graph into quotient graphs. This schema is called modular decomposition of a graph. Linear decomposition algorithms for directed graphs are proposed in [13] and for undirected graph in [14]. As the coloring problem, number of NP-Hard optimization problems find nice solutions if we dispose of a solution for each successive quotient graphs of the decomposition (see [5]).

3 Generalization to hypergraph

The concept of hypergraph corresponding to a set collection has been introduced in a natural way in the sixties to generalize a great number of combinatoric problems defined on graphs.

Definition 5 *([3]) An hypergraph $H = (V, \mathcal{E})$ is a couple of sets where the set V is called a ground set and where $\mathcal{E} = \{E_1, E_2, ..., E_m\}$ a set of subsets of V respects the following :*

1. $E_i \neq \emptyset$ $(i = 1, 2, ..., m)$;
2. $\bigcup_{i=1}^{m} E_i = V$;

Set V corresponds to the vertices of the hypergraph and set $\mathcal{E}$ to its hyperedges. It is clear that a graph corresponds to an hypergraph whose hyperedges are of size two.

A classical problem is the computation of the *k-hyperclique* sets of a *k-hypergraph* H (denoted $HC_k(H)$). A k-hypergraph is an hypergraph with all hyperedges having size k. Let us define formally a k-hyperclique.

Definition 6 *Let $H = (V, \mathcal{E})$ a k-hypergraph, a subset C of V is a k-hyperclique* ***if and only if*** *$\forall E \subseteq C$ such that $|E| = k$ then E belongs to $\mathcal{E}$.*

It is obvious that if $k = 2$, then H is a graph and C is a clique.

An idea to solve this classical problem would be to apply the resolution process described previously for the clique problem. For this we need a twin relation between vertices of H. This relation would permit to reduce the size of the initial instance and to build the whole result with simplicity. Throughout this paper, we will use the k-hyperclique generation problem to illustrate relevance of our twin vertex definitions on hypergraphs.

Since twin vertices are defined using their neighborhood in graphs, we first give a "natural" generalization of the neighborhood of a vertex in hypergraphs :

Definition 7 *Let $H = (V, \mathcal{E})$ and $x \in V$, $\mathcal{V}(x) = \bigcup_{E_i \in \mathcal{E} \text{ and } x \in E_i} E_i \setminus x$ is the neighborhood of vertex x.*

In example given figure 1, with definition 7, we obtain $\mathcal{V}(a) = \{c, d, e\}$ and $\mathcal{V}(b) = \{c, d, e\}$. Thus, since vertices a and b have same neighborhood, they should be considered twin. However a and b do not play a symmetrical role in the set $HC_3(H)$. Indeed, the set $\{a, c, d, e\}$ is a 3-hyperclique of H and thus belongs to $HC_3(H)$. By symmetry, either $\{b, c, d, e\}$ or $\{a, b, c, d, e\}$ should also belong to $HC_3(H)$. That is not the case since hyperedge $\{b, c, e\}$ does not belong to H. As a consequence, the twin definition using the neighborhood as defined in Definition 7 is not interesting since a symmetry in the hypergraph does not imply a symmetry in the generated objects.

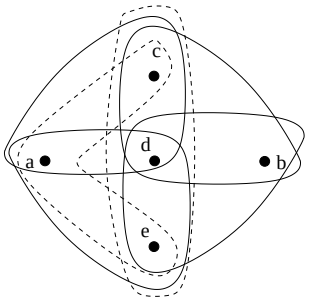

FIG. 1 – 3-Hypergraph H with $\mathcal{E} = \{\{a,c,d\},\{a,c,e\},\{a,d,e\},\{c,d,e\},\{b,c,d\},\{b,d,e\}\}$.

3.1 Clone notion

In the context of collection of sets, authors of [16] have proposed a form of symmetry called clone elements.

Let X be a set of elements of size n and $\mathcal{F}$ a set collection on X. Let x and y be in X. We denote by $\varphi_{x,y} : 2^X \to 2^X$ the mapping which associates to any subset of X its image by swapping items x and y. More formally for any subset F of X we have :

$$\varphi_{x,y}(F) = \begin{cases} (F \setminus \{x\}) \cup \{y\} & if\ y \notin F\ and\ x \in F \\ (F \setminus \{y\}) \cup \{x\} & if\ x \notin F\ and\ y \in F \\ F & otherwise \end{cases}$$

Definition 8 *Let $\mathcal{F}$ be a collection of sets defined on X. Let x and y be in X. We say that x and y are clone in $\mathcal{F}$* ***if and only if*** *for all F in $\mathcal{F}$, we have $\varphi_{x,y}(F)$ in $\mathcal{F}$.*

Note that clone relation defines equivalence classes (cf.[16]). Since hyperedges can be considered as sets, one possible generalization of the twin notion to hypergraphs could be :

Definition 9 *Let $H = (V, \mathcal{E})$ be an hypergraph and x and y be in V. We say that x and y are clone-twin in H* ***if and only if*** *x and y are clone in $\mathcal{E}$.*

Let us check that this definition could be used for our illustration problem. In the example given on figure 1, one can easily check that vertices c and e are clone elements in $\mathcal{E}$. And since $HC_3(H)$ is equal to $H \cup \{a,c,d,e\}$, vertices c and e are also clone elements in $HC_3(H)$. Note also that a and b are not clone elements in $\mathcal{E}$ and thus they are not clone in $HC_3(H)$. Next proposition shows that the clone-twin definition respects the symmetry in the hypergraph as well as in the generated k-hypercliques.

Proposition 1 *Let $H = (V, \mathcal{E})$ be a k-hypergraph and x and y be in V, if x and y are clone vertices in $\mathcal{E}$ then x and y are clone in $HC_k(H)$.*

Contrary to the twin vertices concept defined on the neighbourhood (see definition 7) the concept of clone-twin vertices takes into account the structure of hyperedges. We think that a "good" definition of twin vertices of an hypergraph should reflect strong symmetrical properties on hyperedges. That explains why neighborhood defined twin vertices do not imply a symmetry in the generated objects.

But then, one can question why, in the particular case of graphs, the twin relation defined on the neighborhood of vertices works so well. Main reason is that behind the neighborhood symmetry in graphs lies clone vertices as shown by the following proposition :

Proposition 2 *Let $G = (V, E)$ be a graph and x and y be in V. The following assertions are equivalent :*

- *x et y are twin vertices in G*
- *$V(x) = V(y)$ or $V[x] = V[y]$*
- *x and y are clone in E.*

3.2 Modules and committees in hypergraphs

Another form of symmetry in hypergraphs, based on structural properties of hyperedges, has been defined in term of modules (see [6, 8]).

Definition 10 *Let $H = (V, \mathcal{E})$ be an hypergraph and $M \subseteq V$, we say that M is a module in H* ***if and only if*** *for all A in $\mathcal{E}$ such that $A \cap M \neq \emptyset$ and $A \subseteq M$ we have $\forall P \subseteq M$ such that $P \neq \emptyset$, $(A \setminus M) \cup P$ in $\mathcal{E}$.*

See figure 2 for an example of module in an hypergraph.

We generalize Definition 4 of twins in graphs, using the definition of modules in hypergraphs.

Definition 11 *Let $H = (V, \mathcal{E})$ be an hypergraph and x and y be in V, we say that x and y are module-twin in H* ***if and only if*** *the couple (x, y) makes up a module in H.*

This last proposition constitutes another characterization of twin vertices in hypergraphs. A lesser restrictive generalization of the module concept in hypergraphs is given in ([4, 6, 7]) : the committees.

Definition 12 *Let $H = (V, \mathcal{E})$ be an hypergraph and $C \subseteq V$, we say that C is a committee* ***if and only if*** *for all sets A and B in $\mathcal{E}$ such that $A \cap C \neq \emptyset$ and $B \cap C \neq \emptyset$, we have $(A \cap C) \cup (B \setminus C)$ in $\mathcal{E}$.*

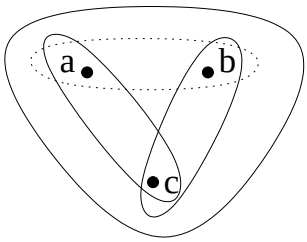

FIG. 2 – Hypergraph $H = (\{a, b, c\}, \{\{a, b\}, \{a, c\}, \{a, b, c\}\})$. The couple (a, b) is a module. Indeed hyperedge $\{a, c\}$ intersects the module and sets $\{a\} \cup \{c\}$, $\{b\} \cup \{c\}$ and $\{a, b\} \cup \{c\}$ belong to the set of hyperedges of H.

To help the understanding we give in figure 3 an example of such a committee. One have to note that if two vertices constitute a module then they constitute a committee.

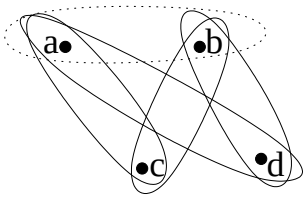

FIG. 3 – Hypergraph $H = (\{a, b, c, d\}, \{\{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}\})$. The couple (a, b) forms a committee since for hyperedge $\{a, c\}$ which intersectes the committee the set $\{b\} \cup \{c\}$ belongs to $\mathcal{E}$, and for hyperedge $\{a, d\}$ which intersect the committee too, the set $\{b\} \cup \{d\}$ belongs to $\mathcal{E}$. (a, b) is not a module since sets $\{a, b, c\}$ and $\{a, b, d\}$ are not hyperedges of H.

Here we propose a new characterization of committee. Let X be a set of elements, C a subset of X and $\mathcal{F}$ a set collection on X. We define a set collection $\mathcal{F}_C^-$ corresponding to the list of sets obtained by computing intersection of each set F from $\mathcal{F}$ with the set C. More formally we have :

– $\mathcal{F}_C^- = \{F \cap C \mid F \in \mathcal{F} \text{ and } F \not\subseteq C \text{ and } C \not\subseteq F\}$;

As the same manner we define a set collection $\mathcal{F}_C^+$ corresponding to the list of sets obtained by computing set difference of each set F from $\mathcal{F}$ with the set C. More formally we have :

– $\mathcal{F}_C^+ = \{F \setminus C \mid F \in \mathcal{F} \text{ and } F \not\subseteq C \text{ and } C \not\subseteq F\}$;

Proposition 3 *Let $\mathcal{F}$ be a collection of sets defined on X, and C a subset of X. We say that C makes up a committee in $\mathcal{F}$* ***if and only if*** *for all F*

in $\mathcal{F}_C^-$ *and for all* F_1 *and* F_2 *in* $\mathcal{F}_C^+$ *with* $F_1 \neq F_2$ *we have* $F \cup F_1 \in \mathcal{F}$ *and* $F \cup F_2 \in \mathcal{F}$.

It seems the concept of committee corresponds to a real kind of symmetry in the hypergraph set of hyperedges. For this reason this concept leads, in a natural way, to a second interesting characterization of twin vertices in hypergraph.

Definition 13 *Let* $H = (V, \mathcal{E})$ *be an hypergraph and* x *and* y *be in* V*, we say that* x *and* y *are committee-twin in* H ***if and only if*** *the couple* (x, y) *makes up a committee in* H.

Committee keeps similar behaviour for the computation of k-hypercliques of a k-hypergraph as it is shown by the following proposition :

Proposition 4 *Let* $H = (V, \mathcal{E})$ *be a k-hypergraph and* x *and* y *be in* V*, if* x *and* y *make up a committee in* $\mathcal{E}$ *then* x *and* y *make up a committee in* $HC_k(H)$.

The following property shows that the **binary** relation "makes up a committee with" between two vertices is also a transitive relation.

Proposition 5 *Let* $H = (V, \mathcal{E})$ *be a hypergraph with vertices* x*,* y *and* z *in* V*, if couples* (x, y) *and* (y, z) *make up committees then the couple* (x, z) *constitutes a committee too.*

Thus, as for clone-twins, committee-twins define equivalence classes. Nevertheless, as we will see in the next section, committee and clone vertices are not equivalent.

3.3 Links between clone-twins and committee-twins

As the clone relation, the **binary** relation "makes up a committee with" infers a partition of an hypergraph vertex set. That means each couple from a same vertex class consitutes a committee. One has to notice we do not say that this class form a whole committee. We just say these classes correspond to equivalence classes of committee-twin vertices. (In the next section we will use these properties of transitivity to compute twin vertices of a given hypergraph). So, the two concepts are not so far from each other. Let us see that clone notion and committee notion are not completely similar.

Let $H = (V, \mathcal{E})$ with $V = \{a, b, c, d\}$ and $\mathcal{E} = \{\{a, c\}, \{b, c\}, \{a, b, d\}\}$, a and b are clone vertices in $\mathcal{E}$ but the couple (a, b) does not form a committee.

Indeed (a, b) is not a committee of H since sets $\{a, d\}$, $\{b, d\}$ and $\{a, b, c\}$ does not belongs to $\mathcal{E}$ (cf. figure 4A). On the same manner consider hypergraph $H = (V, \mathcal{E})$ with $V = \{a, b, c\}$ and $\mathcal{E} = \{\{a, c\}, \{a, b, c\}\}$, couple (a, b) constitutes a committee but vertices a and b are not clone in $\mathcal{E}$ since the set $\{b, c\}$ does not belongs to $\mathcal{E}$ (cf. figure 4B).

FIG. 4 – **A :**$H = (\{a, b, c, d\}, \{\{a, c\}, \{b, c\}, \{a, b, d\}\})$, a and b are clone vertices but the couple (a, b) does not form a committee. **B :** $H = (\{a, b, c\}, \{\{a, c\}, \{a, b, c\}\})$, couple (a, b) constitutes a committee but vertices a and b are not clone.

But if we define simple hypergraph with the following :

Definition 14 *([3]) Let $H = (V, \mathcal{E})$, we say that H is a simple hypergraph* ***if and only if*** *for all sets E and E' in $\mathcal{E}$ we have $E \not\subseteq E'$.*

We have the proposition :

Proposition 6 *Let $H = (V, \mathcal{E})$ be a simple hypergraph and x and y be in V. If the couple (x, y) forms a committee then vertices x and y are clone in $\mathcal{E}$.*

Note that the converse in not true. In other words, in a simple hypergraph committee of size two are a subclass of clone vertices.

In this section we have proposed two characterizations of twin vertices in an hypergraph. One in term of clone vertices (see definition 9) and the other in term of committees (see definition 13). Clone vertices had been characterized by a swapping function (see definition 8). Committees had been characterized with sets $\mathcal{F}_C^-$ and $\mathcal{F}_C^+$ (see proposition 3). Clone binary relation and committee binary relation between vertices induce a partition of the vertex set in twin vertex equivalence classes. Finaly we have shown that, in the general case, clone vertex couples do not correspond to committees. But in the simple hypergraph case, two vertices making up a committee are clone (the reciprocity being false).

4 Algorithms

This section focus on algorithmic aspects of clone and committee concepts described previously. In a first time we give the used data structure corresponding to a mapping. In a second time we present an algorithm to compute clone-twin vertex classes of a given hypergraph. In the third time we give an algorithm to compute committee-twin vertex classes of the hypergraph.

4.1 Abstract Data Type : Key Mapping

In our context we have to manipulate some discret objects as hypergraphs, so we need a data structure that able us to store a set collection. For this we propose to use a Map abstract data type similar to the Map interface of Java language. This data structure maps keys to values. In our case, the keys are the sets of the collection. The values mapped by the keys depend on the algorithm.

This abstract data type supplies the following operators :

- `new()` operator : creates a *Map* object and returns an empty map.
- `get(E)` operator : returns the value associated to the key `E` if this key maps a value, or `Nil` otherwise.
- `put(E,value)` operator : inserts set `E` in the map and associates `value` to it.

Time complexities of those operators deeply rely on the data structure used for the implementation of the Map data type. We propose an implementation which takes advantage of the key type, i.e. sets. To implement the Map type we propose a lexicographic tree (or tries) where sets are represented by branches of the tree. A representation of a hypergraph in term of collection of sets is given figure 5.

Complexities of the `put` and `get` operators rely on the chosen implementation.

Proposition 7 *Let $\mathcal{F}$ be a collection of sets on a ground set X and F be in $\mathcal{F}$, if the lexicographic tree is implemented with `Lists` then :*

- *The `put(F,value)` operator as an $\mathcal{O}(|X|)$ time complexity ;*
- *The `get(F)` operator as an $\mathcal{O}(|X|)$ time complexity ;*

Access complexity is due to the fact that an element of X appears only once in a set and that there exists an ordering on X. Cost of a node creation is done in constant time. See [1] for more information about the trie implementing and performance aspects.

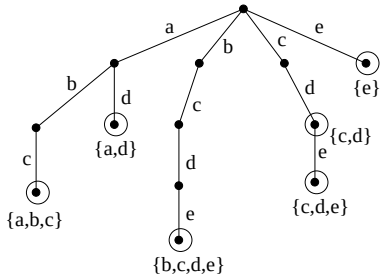

FIG. 5 – The lexicographic tree corresponding to an hypergraph $H = (V, \mathcal{E})$ with $V = \{a, b, c, d, e\}$ and $\mathcal{E} = \{\{a, b, c\}, \{a, d\}, \{b, c, d, e\}, \{c, d\}, \{c, d, e\}, \{e\}\}$. We suppose existence of an ordering on vertices of V. In that case the value corresponding to a key is a boolean. This boolean is true (circled nodes) if the node corresponds to an hyperedges and false if the node corresponds to a intermediary node.

4.2 Clone-twin classe computation

We propose here, in a compacter form, the algorithm given in [17]. For a set collection $\mathcal{F}$ on a ground set X, authors introduce the distance notion between elements of X. By this way they give a new characterization of clone elements in term of distance between them. In term of hypergraph we obtain the following characterization.

Definition 15 *Let $\mathcal{E}$ be a set of hyperedges on V a set of vertices. Let x and y be in V. We call distance between x and y, denoted by $d_{\mathcal{E}}(x, y)$, the mapping :*

$$\begin{aligned} V^2 &\rightarrow N \\ d_{\mathcal{E}}(x, y) &\rightarrow |\{E \in \mathcal{E} \mid \varphi_{x,y}(E) \notin \mathcal{E}\}| \end{aligned}$$

In other words, the distance between vertices x and y represents the number of hyperedges E containing x XOR y such its image $\varphi_{x,y}(E)$ is not in $\mathcal{E}$. One can notice this mapping is symmetrical and it is bounded by the number of hyperedges containing x but y (denoted $|\mathcal{E}_{x\overline{y}}|$) plus the number of hyperedges containing y but x (denoted $|\mathcal{E}_{\overline{x}y}|$). From definition 15 we have the following porperty.

Proposition 8 ([17])
Let $\mathcal{E}$ be a set of hyperedges on V and x and y in V, x and y are clone vertices in $\mathcal{E}$ ***if and only if*** *$d_{\mathcal{E}}(x, y) = 0$.*

All the algorithmic process proposed in [17] is supported by the following proposition. It characterizes a couple (E, E') from the set $\mathcal{E}^2$ such that $E = \varphi_{x,y}(E')$.

Proposition 9 *Let $\mathcal{E}$ be a set of hyperedges defined over V, E and E' two distinct hyperedges of $\mathcal{E}$ and (x, y) a couple of vertices of V such that $x \in E$ and $y \in E'$. Then the following assertions are equivalent :*

1. $E = \varphi_{x,y}(E')$
2. $E' = \varphi_{x,y}(E)$
3. $|E| = |E'|$ *and* $E \setminus E' = \{x\}$ *and* $E' \setminus E = \{y\}$
4. $E \setminus \{x\} = E' \setminus \{y\}$

This proposition states that two hyperedges E and E' are their respective images by the swapping function φ **if and only if** they have same size t and share $t - 1$ vertices. This property follows directly from the definition of the φ mapping.

From this proposition a process in three steps is described : in a first step we initialize the distance matrix (cf. algorithme 1). In a second step we compute the distance for each vertex couple (cf. algorithme 2). In the last step we deduce from the matrix the clone vertex classes of an hyperedge set (cf. algorithm 3).

4.2.1 Initializing the distance matrix

We initialize each distance $d_{\mathcal{E}}(x, y)$ with its maximal possible value, i.e. with $|\mathcal{E}_{x\overline{y}}| + |\mathcal{E}_{\overline{x}y}|$. To do that we intialize all the distances to 0 and then, for each $E \in \mathcal{E}$ we increment by 1 the distances $d_{\mathcal{E}}(x, y)$, with $x \in E$ and $y \notin E$ (cf. algorithm 1). This can be done in $\mathcal{O}(|V| \times ||\mathcal{E}||)$. Note that $||\mathcal{E}||$ corresponds to the sum of the size of each hyperedge of $\mathcal{E}$.

4.2.2 Distance algorithm

This algorithm relies on Property 9. Let consider E, E' and E'' be hyperedges of $\mathcal{E}$ such that $E = \varphi_{x,y}(E')$ and $E = \varphi_{x,z}(E'')$. Then, according to Property 9 we have $E \setminus \{x\} = E' \setminus \{y\} = E'' \setminus \{z\}$. And thus, $E' = \varphi_{y,z}(E'')$. Idea of the algorithm is to compute classes of sets E_i of $\mathcal{E}$ having $|E_i| - 1$ common items. Thus, a class C can be represented by the set of common items and we memorize in a set `Union` all the extra items x_i which are not common. In the Map structure we use, the set C will be the key while the set `Union` will be the value associated to the key.

Algorithm 1: $InitDistance(\mathcal{E})$

Data : A set of hyperedges $\mathcal{E}$ defined over V.
Result: The distance matrix $d_{\mathcal{E}}$ such that for all x and y in V we have $d_{\mathcal{E}}(x,y) = |\mathcal{E}_{x\overline{y}}| + |\mathcal{E}_{\overline{x}y}|$.

```
begin
    foreach (x ∈ V) do
        foreach (y ∈ V) do
            d_E(x, y) = 0 ;
        end
    end
    foreach (E ∈ E) do
        foreach (x ∈ E) do
            foreach (y ∉ E) do
                d_E(x, y) + + ;
            end
        end
    end
    return d_E ;
end
```

Then, for any $E \in C$ and for any $(x,y) \in$ `Union`, we know that, according to Property 9, we have $\varphi_{x,y}(E) \in \mathcal{E}$ and $E \neq \varphi_{x,y}(E)$. And thus, $d_{\mathcal{E}}(x,y)$ has to be decremented. Note that a set E can belong to at most $|E|$ classes.

The algorithm 2 is quite straightforward using the Map structure. For all sets E of $\mathcal{E}$ we insert each of its $|E|$ subsets of size $|E|-1$ in the Map structure. If the key was already present, we just append the extra item of E to the set `Union` and update all the necessary entries in the distance matrix. Otherwise, a new key $E \setminus \{x\}$ is present in the Map structure and its associated `Union` value is initialized with $\{x\}$.

Proposition 10 ([17])
Let $\mathcal{E}$ be a set collection on a ground set V, algorithm 2 computes the distance matrix $d_{\mathcal{E}}$ in $\mathcal{O}(|V| \times ||\mathcal{E}||)$ time complexity in the worst case.

4.2.3 Clone-twin classe algorithm

The Computation of the clone-twin classes from the distance matrix corresponds to the last step of the process. First, let us recall that two vertices x and y are clone **if and only if** $d_{\mathcal{E}}(x,y) = 0$. Since, the clone relation is an equivalence relation, it defines a partition of the set V. Principle of algorithm 3 is the following. Let x be a vertex of V which has still not been assigned to a class. We then search all remaining elements y which distance with x is

Algorithm 2: *Distance*($\mathcal{E}$)

Data : A set of hyperedges $\mathcal{E}$ defined over V.
Result: The distance matrix $d_{\mathcal{E}}$.
begin
  $d_{\mathcal{E}} = InitDistance(\mathcal{E})$;
  $\mathcal{T}$ = `new Map()` ;
1 **foreach** $(E \in \mathcal{E})$ **do**
2 **foreach** $(x \in E)$ **do**
  $C = E \setminus \{x\}$
3 `Union` = $\mathcal{T}$.`get`(C)
  if *Union* $\neq$ *Nil* **then**
4 **foreach** $(y \in$ *Union*$)$ **do**
  $d_{\mathcal{E}}(x, y) = d_{\mathcal{E}}(x, y) - 2$; $d_{\mathcal{E}}(y, x) = d_{\mathcal{E}}(y, x) - 2$
  end
  `Union` = `Union` $\cup\{x\}$;
  $\mathcal{T}$.`put`($C, Union$) ;
  else
5 $\mathcal{T}$.`put`($C, \{x\}$)
  end
  end
  end
  return $d_{\mathcal{E}}$;
end

null. All those vertices will form a clone class with x and thus are removed from the list of vertices which are not assigned to a class. The class of x is then stored in a list $\mathcal{L}$.

Proposition 11 *Let $\mathcal{E}$ be a set collection on a ground set V, algorithm 3 computes clone-twin classes of $\mathcal{E}$ in $\mathcal{O}(|V| \times ||\mathcal{E}||)$ time and space complexities in the worst case.*

Algorithm 3: $Clone - TwinClasses(\mathcal{E})$

```
   Data  : A set of hyperedges E defined over V.
   Result: The list L of clone classes.
   begin
1      d_E = Distance(E);
       L = ∅; temp = V;
       while (temp ≠ ∅) do
2          foreach (x ∈ temp) do
               l_x = newList();
               l_x = l_x ∪ {x};
               temp = temp \ {x};
3              foreach (b ∈ temp) do
                   if (d_E(x, y) = 0) then
                       l_x = l_x ∪ {y};
                       temp = temp \ {y};
                   end
               end
               L = L + l_x;
           end
       end
       return L;
   end
```

4.3 Committee-twin classe computation

In this subsection we prosose two algorithms. A first one, algorithm 4, acknowledges a set as committee. From this algorithm we give a second one, algorithm 5, to compute the committee-twin classes from a given hyperedge set.

4.3.1 Acknowledge a set as committee algorithm

Principle of algorithm 4 is to insert in a whole map a set of keys corresponding to the set $\mathcal{E}_C^-$. To each of these keys E^- we associate as value the

list of sets E^+ such that $E^- \cup E^+$ belongs to $\mathcal{E}$. The algorithm returns *true* **if and only if** the list of sets associated to each key is identical.

Algorithm 4: IsACommittee ?($\mathcal{E}$,C)

Data : A set of hyperedges $\mathcal{E}$ on a ground set V, a set $C \subseteq V$.
Result: True **if and only if** C is a committee.
begin
  $\mathcal{T}$ = `new Map()` ;
1 **foreach** $(E \in \mathcal{E})$ **do**
    $E^- = E \cap C$ and $E^+ = E \setminus C$;
    if $[(E^+ \neq \emptyset)$ *and* $(E^- \neq \emptyset)]$ **then**
      $collection = \mathcal{T}.get(E^-)$;
2 **if** $(E^+ \notin collection)$ **then**
        $collection = collection \cup E^+$
      end
      $\mathcal{T}.put(E^-, collection)$
    end
  end
  $returnValue = true$;
  $collection = \mathcal{T}.firstValue()$;
3 **if** $(|collection| = 1)$ **then**
    return $false$;
  else
4 **foreach** $(E \in \mathcal{T})$ **do**
      if $(collection \neq \mathcal{T}.get(E))$ **then**
        $returnValue = False$;
      end
    end
    return $returnValue$;
  end
end

Proposition 12 *Let $\mathcal{E}$ be a set collection on a ground set V, and $C \subseteq V$, algorithme 4 aknowledges set C as committee in $\mathcal{O}(|V| \times |\mathcal{E}|)$ time complexity in the worst case.*

4.3.2 Committee-twin classe algorithm

With the same principle than algorithm 3 the following alogrithm 5 computes committee-twin classes.

Algorithm 5: $Committee - TwinClasses(\mathcal{E})$

```
   Data  : A set of hyperedges E defined over V.
   Result: The list L of clone classes.
   begin
       L = ∅ ; temp = V ;
       while (temp ≠ ∅) do
1          foreach (x ∈ temp) do
               l_x = newList() ;
               l_x = l_x ∪ {x} ;
               temp = temp \ {x} ;
2              foreach (b ∈ temp) do
3                  if (IsACommittee?(E, {x, y}) then
                       l_x = l_x ∪ {y} ;
                       temp = temp \ {y} ;
                   end
               end
               L = L + l_x ;
           end
       end
       return L ;
   end
```

Proposition 13 *Let $\mathcal{E}$ be a set collection on a ground set V, algorithme 5 computes committee-twin classes in $\mathcal{O}(|V|^3 \times |\mathcal{E}|)$ time complexity in the worst case.*

In this section we have proposed two algorithms to compute twin vertex classes for a given hyperedge set. Algorithm 3 which computes clone-twin classes and algorithm 5 which computes committee-twin classes. Their principle is the same. The first one has to test if two vertices are clone by cheking their distance (cf. algorithm 2), the second one has to test if two given vertices makes up a committee by using $IsACommittee()$ algorithm (cf. algorithm 4).

5 Conclusion

It is well known (an used) that numerous generation problems on graphs can be reduced to equivalent problems on graphs without twin vertices. When dealing with hypergraphs, very few reduction schemas are used. To our knowledge, only reduction to *simple* hypergraphs is widely used. In [16], authors propose a reduction process based on clone vertices. Main contribution of this

paper is to extend the notion of twin vertices to hypergraphs. We are deeply convinced that the twin vertices definition should rely on *symmetries* over hyperedges which are preserved on the generated objects. This implies that the twin vertex notion is intrinsically depending on the studied generation problem. This lead to the following very general definition of twin vertices.

Definition 16 *Let $H = (V, \mathcal{E})$ be an hypergraph, P a generation problem over H and* Sym *a symmetry property. Vertices x and y of H are said to be* Sym-twins **if and only if** $\mathrm{Sym}_{x,y}(H)$ *and* $\mathrm{Sym}_{x,y}(P(H))$[3] *are true.*

In this paper, we show that symmetries based on clone vertices and on committees are consistent with this definition. Moreover, the definition of twin vertices in the particular case of graphs is simply a clone symmetry in 2-hypergraphs. Note that according to this definition, new symmetries could be found on graphs and thus lead to new definitions of twin vertices in graphs.

In this paper, we provide two algorithms which computes twin vertex classes. The first one is based on the clone symmetry and has an $O(|V| \times ||\mathcal{E}||)$ time complexity. The second relies on committees and has an $O(|V|^3 \times |\mathcal{E}|)$ time complexity. Note that, in this later case, we based the algorithm on a committee recognition algorithm (committees might have any size for this algorithm). A more particular algorithm for the committee-twin vertex class detection should enhance the time complexity.

We showed that for *simple hypergraphs*, committee-twin vertices are also clone-twin vertices (the converse is not true). Thus, when dealing with simple hypergraphs, using the clone symmetry property is sufficient, more general and more efficient.

Given a generation problem P on hypergraph H, the overall strategy should be the following :

- Check if there exists a symmetry property *Sym* on H and $P(H)$ (obviously, one should first verify if the symmetries based on clone vertices and committees are verified) ;
- In the affirmative, the next step is to reduce the hypergraph H to H' according to some reduction algorithm (which might be specific to the studied generation problem) ;
- Generate $P(H')$ using a classical generation algorithm ;
- Reconstruct the missing object (i.e. $P(H) \setminus P(H')$) using some reconstruction algorithm (which depends on the reduction algorithm).

[3] $Sym_{x,y}(H)$ means that x and y verify the symmetry property on H. $P(H)$ represents the collection of objects generated over H using an algorithm wich fulfills the generation problem P.

Note that during the reduction process, several symmetries can be combined (this should be memorized for the reconstruction). In some cases, the reduction process could lead to hypergraphs with new classes of twin vertices which where not twin in the original hypergraph. And thus, this reduction process could be repeated until no more twin vertices are found (for all the symmetry properties considered).

6 Appendice

Proposition 1 *Let $H = (V, \mathcal{E})$ be a k-hypergraph, and x and y in V, if x and y are clone vertices in $\mathcal{E}$ then x and y are clone in $HC_k(H)$.*

Proof : Let $H = (V, \mathcal{E})$ be a k-hypergraph and x and y clone in $\mathcal{E}$. We have to show that for all C in $HC_k(H)$, $\varphi_{x,y}(C)$ belongs to $HC_k(H)$. Without lost of generality let us suppose that x belongs to C and y does not belong to C (indeed, in other cases $C = \varphi_{x,y}(C)$ and then $\varphi_{x,y}(C)$ belongs to $HC_k(H)$). In other words we have to show that $\forall E \subseteq \varphi_{x,y}(C) = C \setminus x \cup y$, $E \in \mathcal{E}$ since $\varphi_{x,y}(C)$ must be an hyperclique.

Different cases occur :

1. If $y \notin E$ then $E \subset C$ and thus $E \in \mathcal{E}$ since C is an hyperclique.
2. If $y \in E$ then $\varphi_{x,y}(E) \subseteq C$ and thus $\varphi_{x,y}(E) \in \mathcal{E}$ since C is a k-hyperclique. $E = \varphi_{x,y}(\varphi_{x,y}(C))$ belongs to $\mathcal{E}$ since x and y are clone in $\mathcal{E}$.

Thus, if a given set C is an hyperclique then $\varphi(C)$ is also an hyperclique.

Proposition 2 *Let $G = (V, E)$ be a graph and x and y be in V. The following assertions are equivalent :*

1. *x et y are twin vertices in G*
2. *$V(x) = V(y)$ or $V[x] = V[y]$*
3. *x and y are clone in E.*

Proof :

- $1 \Leftrightarrow 2$ by definition of twin vertices ;
- $1 \Rightarrow 3$: let us show that if a et b are twin vertices in G then $\forall (x, y) \in E$ then $\varphi_{a,b}(x, y)$ belongs to E. Whithout loss of generality let us consider following cases. If $(x, y) = (a, b)$ (in that case a and b are true twins) then $\varphi_{a,b}(a, b) = (a, b)$ which belongs to E. If $x = a$ and $y \neq b$ then $\varphi_{a,b}(a, y) = (b, y)$ which belongs to E since $V(x) = V(y)$. If $x \neq a$ and $y \neq b$ then $\varphi_{a,b}(x, y) = (x, y)$ (by definition of $\varphi_{a,b}$) which belongs to E.

– $3 \Rightarrow 2$: let us suppose that x and y are clone in E, then $\forall (a, x) \in E$, (b, x) belongs to E. That means $V(x) = V(y)$. Two cases occure. If (a, b) does not belong to E then a and b are false twins. On the contrary, a and b are true twins and $V[x] = V[y]$.

Proposition 4 *Let $H = (V, \mathcal{E})$ be a k-hypergraph, and x and y in V, if x and y make up a committee in $\mathcal{E}$ then x and y make up a committee in $HC_k(H)$.*

Proof : Class of k-hypergraph is a subclass of simple hypergraph. We have shown with proposition 6 that two vertices making up a committe in simple hypergraph are clone vertices.

Proposition 5 *Let $H = (V, \mathcal{E})$ be a hypergraph with vertices x, y and z in V, if couples (x, y) and (y, z) make up committees then the couple (x, z) constitutes a committee too.*

Proof :

Lemma 1 *Let $H = (V, \mathcal{E})$ be a hypergraph with items x, y and z in V, if sets $\{x, y\}$ and $\{y, z\}$ make up committees then $\forall e \in \mathcal{E}$ of the form xp with $p \subseteq V$ and $\{z\} \notin p$ then $\{z\} \cup p \in \mathcal{E}$).*

Proof : Two cases occure :

1. let us supppose $\{y\} \notin p$ then $p \cup \{y\} \in \mathcal{E}$ since $\{x, y\}$ is a committee and then $p \cup \{z\} \in \mathcal{E}$ since $\{y, z\}$ is a committee. Thus the set $\{z\} \cup p$ belongs to $\mathcal{E}$.
2. let us suppose $\{y\} \in p$ then $p \cup \{x\} = p' \cup \{x, y\} \in \mathcal{E}$ implies $p' \cup \{x, z\} \in \mathcal{E}$ since $\{y, z\}$ is a committee and then $p' \cup \{y, z\} \in \mathcal{E}$ since $\{x, y\}$ is a committee. Thus $p \cup \{z\}$ belongs to $\mathcal{E}$.

Let us suppose that the set $\{x, z\}$ does not make up a committee then $\exists e \in \mathcal{E}$ of the form $p \cup \{x\}$ such that $z \notin p$ and $p \cup \{z\} \notin \mathcal{E}$. This is in contradiction with the previous lemma.

Proposition 6 *Let $H = (V, \mathcal{E})$ be a simple hypergraph and x and y be in V. If the couple (x, y) forms a committee then vertices x and y are clone in $\mathcal{E}$.*

Proof : Let $H = (V, \mathcal{E})$ be a simple hypergraph and x and y are not clone vertices in $\mathcal{E}$. So there exists E in $\mathcal{E}$ such that $\varphi_{x,y}(E) \notin \mathcal{E}$. Let us suppose this set contains x but y, then we have $(E \backslash x) \cup y \notin \mathcal{E}$, and $(E \backslash x) \cup \{x, y\} \notin \mathcal{E}$ since H is a simple hypergraph (we recall that E belongs to $\mathcal{E}$). So $\{x, y\}$ could not be a committee since the set $\{y\} \cap \{x, y\} \cup p \setminus \{x\} = p \setminus \{x\} \cup \{y\} \notin \mathcal{E}$.

Proposition 10 *Let $\mathcal{E}$ be a set collection on a ground set V, algorithm 2 computes the distance matrix $d_{\mathcal{E}}$ in $\mathcal{O}(|V| \times ||\mathcal{E}||)$ time complexity in the worst case.*

Proof : Correctness of algorithm 2 comes from property 9. Initialization of the distance matrix is done in $\mathcal{O}(|V| \times ||\mathcal{E}||)$. We suppose that the Map structure is implemented using a lexicographic tree with `Lists`. Loops in line 1 and 2 do together $||\mathcal{E}||$ iterations. In line 3, the retrieval is done in $\mathcal{O}(|V|)$ thanks to the lexicographic tree. The loop in line 4 does at most $|V|$ iterations and the update of the matrix takes constant time. The insertion of line 5 is done in $\mathcal{O}(|V|)$. Thus, the overall complexity is in $\mathcal{O}(\sum_{E \in \mathcal{E}} |E| \times |V|) = \mathcal{O}(|V| \times ||\mathcal{E}||)$.

Proposition 11 *Let $\mathcal{E}$ be a set collection on a ground set V, algorithm 3 computes clone-twin classes of $\mathcal{E}$ in $\mathcal{O}(|V| \times ||\mathcal{E}||)$ time and space complexities in the worst case.*

Proof : Correctness of the algorithm 3 comes from property 8. Line 1 is in $\mathcal{O}(|V| \times ||\mathcal{E}||)$. Loops in line 2 and 3 do together $|V^2|$ operations. The whole process is then in $\mathcal{O}(|V| \times ||\mathcal{E}||)$.

Proposition 12 *Let $\mathcal{E}$ be a set collection on a ground set V, and $C \subseteq V$, algorithme 4 aknowledge set C as committee in $\mathcal{O}(|V| \times |\mathcal{E}|)$ time complexity in the worst case.*

Proof : Loop in line 1 build a whole map such that the set of keys corresponds to the set $\mathcal{E}_C^-$. To each key is associated a collection of sets. From property 8 we know that C is a committee **if and only if** each collection is identical (the test is made with the loop in line 4). Indeed, in that case, this collection is equal to $\mathcal{E}_C^+$. Moreover to define a committee we need to have at least two hyperedges which intersect it. For this reason a test is done in line 3 on the size of the collection. The size of the first loop in line 1 is $|\mathcal{E}|$. By using a map structure to manage the collection all the following instructions can be done in $\mathcal{O}(|V|)$. The loop in line 4 tests if each key is associated to the same collection. If we suppose that each collection is stored in a lexicographique tree then the whole process of this test can be done in $\mathcal{O}(||\mathcal{E}||)$. Thus algorithm 4 aknowledges set C as committee in $\mathcal{O}(|V| \times |\mathcal{E}|)$ time complexity in the worst case.

Proposition 13 *Let $\mathcal{E}$ be a set collection on a ground set V, algorithme 5 computes committee-twin classes in $\mathcal{O}(|V|^3 \times |\mathcal{E}|)$ time complexity in the worst case.*

Proof : Principle of algorithm 5 is to test if each couple of vertices from V forms a committee. From property 12 we know this test can be done in $\mathcal{O}(|V| \times |\mathcal{E}|)$. Thus the whole process is in $\mathcal{O}(|V|^3 \times |\mathcal{E}|)$ time complexity in the worst case.

Références

[1] S. Bachelard, O. Raynaud, and Y. Renaud. Implementing sets collection with trie : a stepping stone to performances ? *Research Report : L.I.M.O.S. RR-06-06*, 2006.

[2] H.J. Bandelt and H.M. Mulder. Distance-hereditary graphs. pages 41 :182–208, 1986.

[3] C. Berge. *Hypergraphes*. Gauthier-villars, 1987.

[4] L.J. Billera. On the composition and the decomposition of clutters. B,11 :234–245, 1971.

[5] B. Bollobas. Extremal graph theory. 1978.

[6] M. Chen, M. Habib, and M.C. Maurer. Partitive hypergraphs. 37 :35–50, 1981.

[7] W.H. Cunningham and J. Edmonds. A combinatorial decomposition theory. 32,3 :734–765, 1980.

[8] A. Ehrenfeucht and R. McConnell. A k-structure generalization of the theory of 2-structures. 132 :209–227, 1994.

[9] M. R. Garey and D.S. Johnson. *Computers and Intractability*. W.H. Freeman and Company, 1979.

[10] A. Gely, R. Medina, L. Nourine, and Y. Renaud. Uncovering and reducing hidden combinatorics in guigues-duquenne covers. In *ICFCA'05*, 2005.

[11] M. Habib, L. Nourine, O. Raynaud, and E. Thierry. Computational aspects of the two dimension of partially ordered sets. *Theoritical Computer Science*, 312(2-3) :401–431, 2004.

[12] J.M. Lanlignel, O. Raynaud, and E. Thierry. Pruning graphs with digital search trees. application to distance hereditary graphs. In *17th Annual Symposium on Theoritical Aspects of Computer Science*, pages 529–541, 2000.

[13] R.M. McConnell and F. de Montgolfier. Linear-time modular decomposition of directed graphs. pages 189–209, 2005.

[14] R.M. McConnell and J.P. Spinrad. Linear-time modular decomposition and efficient transitive orientation of comparability graphs. In *5th ACM-SIAM Symposium on Discrete Algorithms, SODA'94*, pages 536–545, 1994.

[15] R. Medina and L. Nourine. Algorithme efficace de generation des ideaux d'un ensemble ordonne.

[16] R. Medina and L. Nourine. Clone items : a pre-processing information for knowledge discovery. *submitted*.

[17] R. Medina, C. Noyer, and O. Raynaud. Efficient algorithms for clone items detection. In *CLA'05*, pages 70–81, 2005.

[18] L. Nourine and O. Raynaud. A fast algorithm for building lattices. *Information Processing Letters*, volume 71 :199–204, 1999.